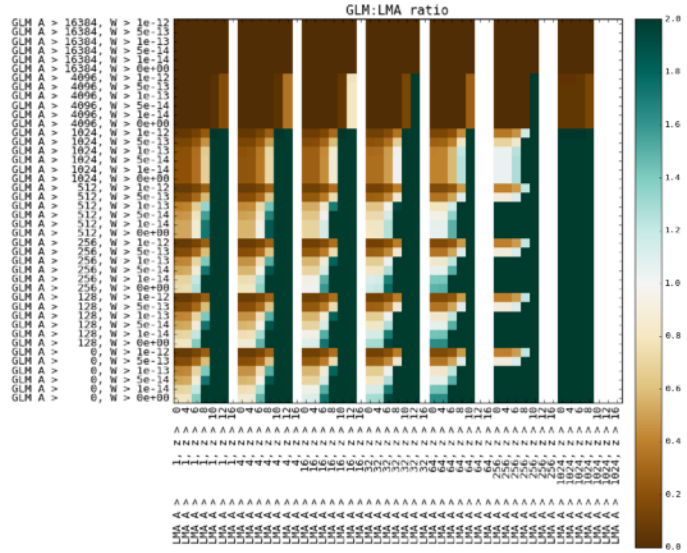
TEXAS TECH UNIVERSITY™

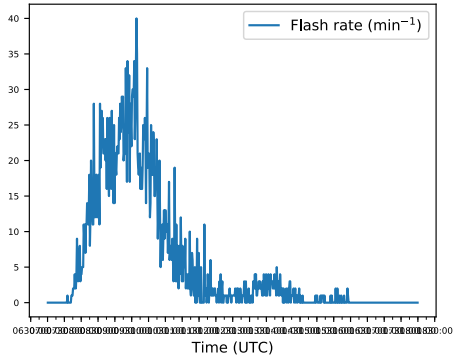
A Python GLM reader supporting flash gridding and time series analyses using lmatools

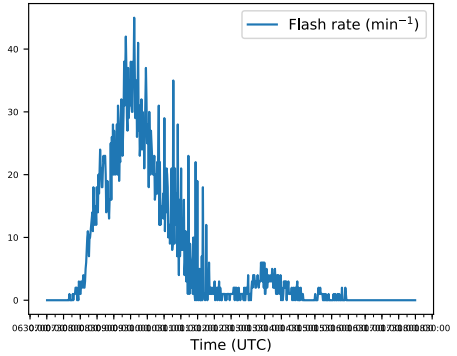
Eric Bruning
TTU Department of Geosciences
Atmospheric Science Group

GLM Science Meeting, Huntsville, AL
12-14 September, 2017

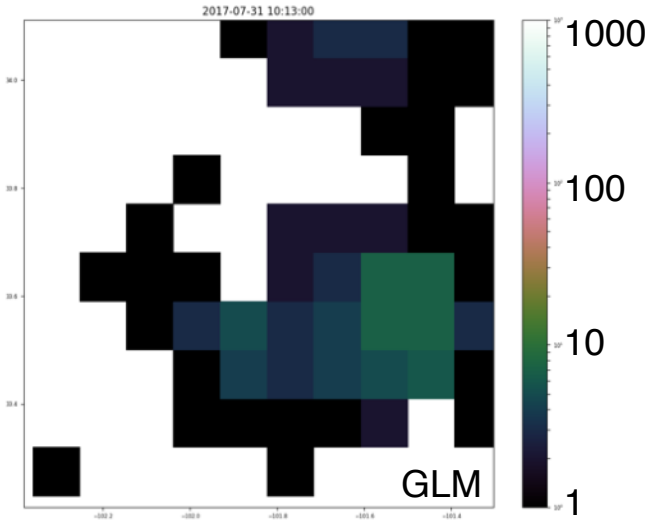
Parameter space study and useful diagnostics from one shell script

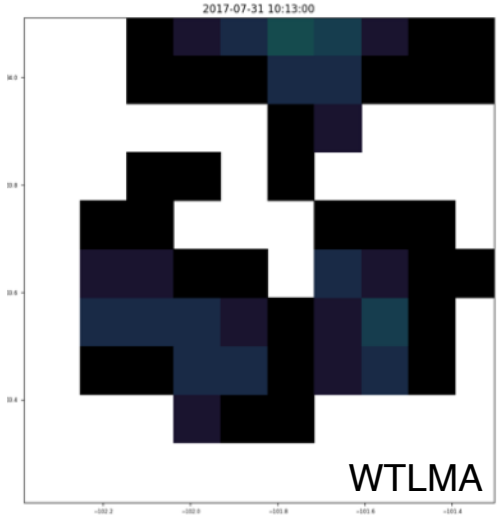






Flash extent density, 1013 UTC



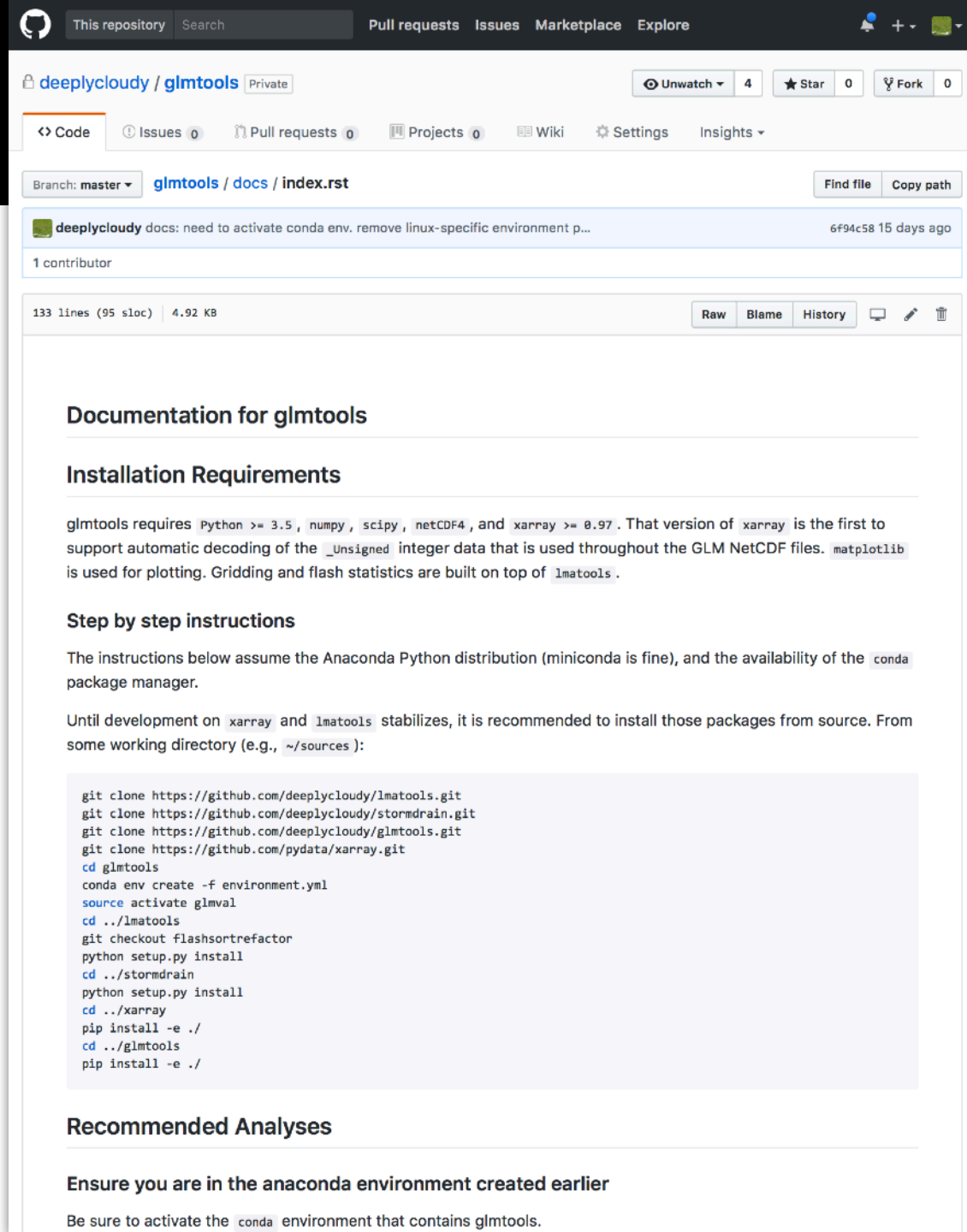



```
#!/bin/bash
export MPLBACKEND="Agg"

export GLMGRIDSCRIPT=~/sources/glmtools/examples/grid/make_GLM_grids.py
export GLMLASSOSCRIPT=~/sources/glmtools/examples/glm-lasso-stats.py
export LMALASSOSCRIPT=~/sources/lmatools/examples/lasso/cell-lasso-stats.py

export GLMSORTGRID=./GLM/
export LMASORTGRID=./LMA/

export LASSO=lasso-WTLMA-50km.txt

export CTRLAT=33.5
export CTRLON=-101.5
export WIDTH=800

export GLMFILES="/archive/GLM/GLM-L2-LCFA_G16/2017/Jul/31/OR_GLM-L2-LCFA_G16_s201721
export LMAFILES="/archive/GLM/LMA/h5_files/2017/Jul/31/LYLOUT_170731_0[7-9]*.h5 /arc

echo "Processing LMA flashes to grid"
python $GLMGRIDSCRIPT \
    -o $LMASORTGRID/grid_files/ \
    --nevents 10 \
    --ctr_lat $CTRLAT --ctr_lon $CTRLON --width $WIDTH --height $WIDTH \
    --lma $LMAFILES

echo "Processing GLM flashes to grid"
for GLMEVENTS in 1 2 4
do
    python $GLMGRIDSCRIPT \
        -o $GLMSORTGRID/minevent$GLMEVENTS/grid_files/ \
        --nevents $GLMEVENTS \
        --ctr_lat $CTRLAT --ctr_lon $CTRLON --width $WIDTH --height $WIDTH \
        $GLMFILES &
done

wait

for LMAAREA in 1 4 16 32 64 256 1024
# all flashes with greater than this area
do
    for LMAALT in 0 4 6 8 10 12 16
    # all flashes whose centroid is greater than each of these altitudes
    do
        echo "Processing LMA flashes for area$LMAAREA_energy$LMAENERGY"
        python $LMALASSOSCRIPT -l $LASSO -s $LMASORTGRID --skipspectra \
            --minarea $LMAAREA --minalt $LMAALT \
            -o LMA_area$LMAAREA\_ctralt$LMAALT \
            $LMAFILES &
    done
    wait
done

for GLMEVENTS in 1 2 4
# Use GLM grids that have been filtered by min events before producing
do
    for GLMAREA in 0 128 256 512 1024 4096 16384
    # all flashes greater than this area
    do
        for GLMENERGY in 0 "1.0e-14" "5.0e-14" "1.0e-13" "5.0e-13" "1.0e-12"
        # all flashes larger than this energy/radiance
        do
            python $GLMLASSOSCRIPT --skip3d -l $LASSO --skipspectra \
                -s $GLMSORTGRID/minevent$GLMEVENTS/ \
                --nevents $GLMEVENTS --minarea $GLMAREA --minenergy $GLMENERGY \
                -o GLM_events$GLMEVENTS\_area$GLMAREA\_energy$GLMENERGY \
                $GLMFILES &
        done
        wait
    done
done
```

BASIC FEATURES

- GLM reader: Built-in traversal of FGE hierarchy
  - *Includes converter from Mach LCFA ASCII to NetCDF*
- Gridding to lat/lon, map projection, or fixed grid
- Time series trends of flash rate and properties
  - *Within latitude / longitude box or moving cell lasso*

- Built on lmatools, so can perform parallel analyses of both LMA and GLM data
- Improved usability: command line scripts with parameterized configuration (e.g. `--ngroups 2`)



This repository Search Pull requests Issues Marketplace Explore

deeplycloudy / glmtools Private

Unwatch 4 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Settings Insights

Branch: master glmtools / docs / index.rst Find file Copy path

deeplycloudy docs: need to activate conda env. remove linux-specific environment p... 6f94c58 15 days ago

1 contributor

133 lines (95 sloc) 4.92 KB Raw Blame History

Documentation for glmtools

Installation Requirements

glmtools requires `Python >= 3.5`, `numpy`, `scipy`, `netCDF4`, and `xarray >= 0.97`. That version of `xarray` is the first to support automatic decoding of the `_Unsigned` integer data that is used throughout the GLM NetCDF files. `matplotlib` is used for plotting. Gridding and flash statistics are built on top of `lmatools`.

Step by step instructions

The instructions below assume the Anaconda Python distribution (miniconda is fine), and the availability of the `conda` package manager.

Until development on `xarray` and `lmatools` stabilizes, it is recommended to install those packages from source. From some working directory (e.g., `~/sources`):

```
git clone https://github.com/deeplycloudy/lmatools.git
git clone https://github.com/deeplycloudy/stormdrain.git
git clone https://github.com/deeplycloudy/glmtools.git
git clone https://github.com/pydata/xarray.git
cd glmtools
conda env create -f environment.yml
source activate glmval
cd ../lmatools
git checkout flashsortrefactor
python setup.py install
cd ../stormdrain
python setup.py install
cd ../xarray
pip install -e ./
cd ../glmtools
pip install -e ./
```

Recommended Analyses

Ensure you are in the anaconda environment created earlier

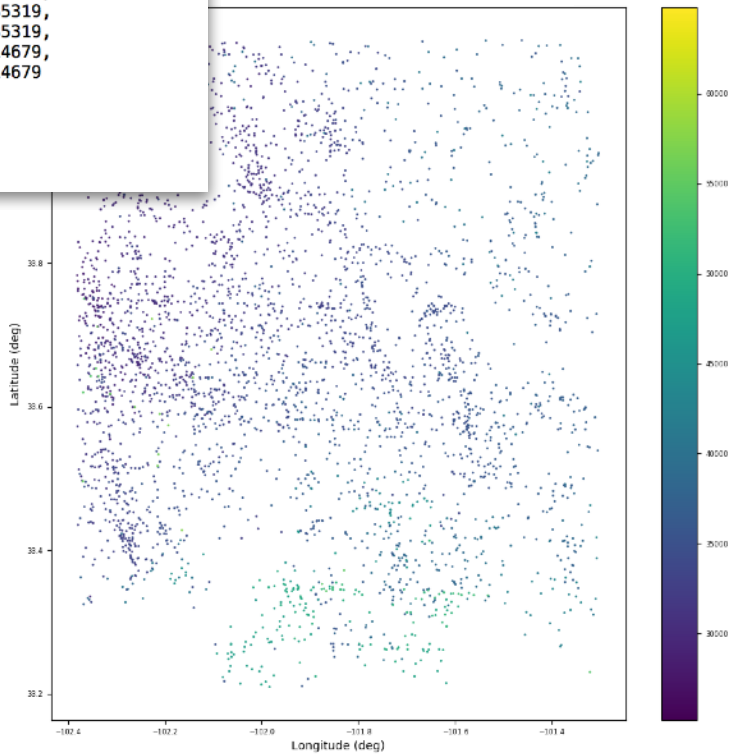
Be sure to activate the `conda` environment that contains glmtools.

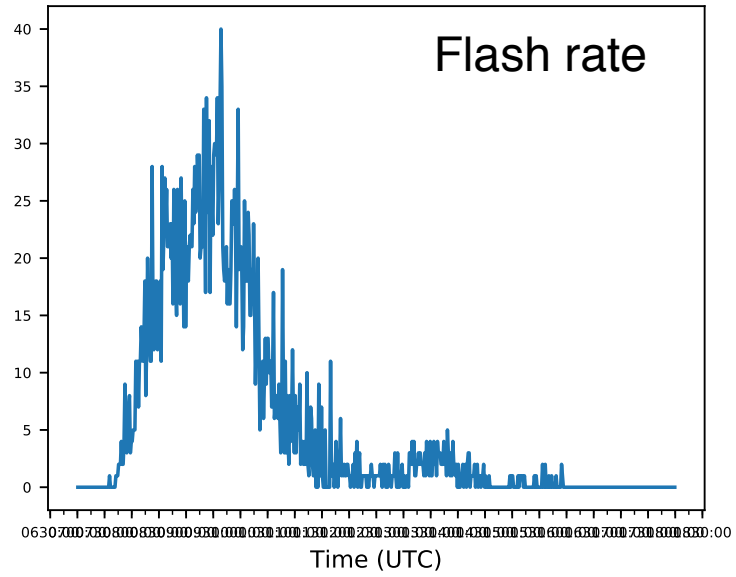
Flash statistics within bounding box

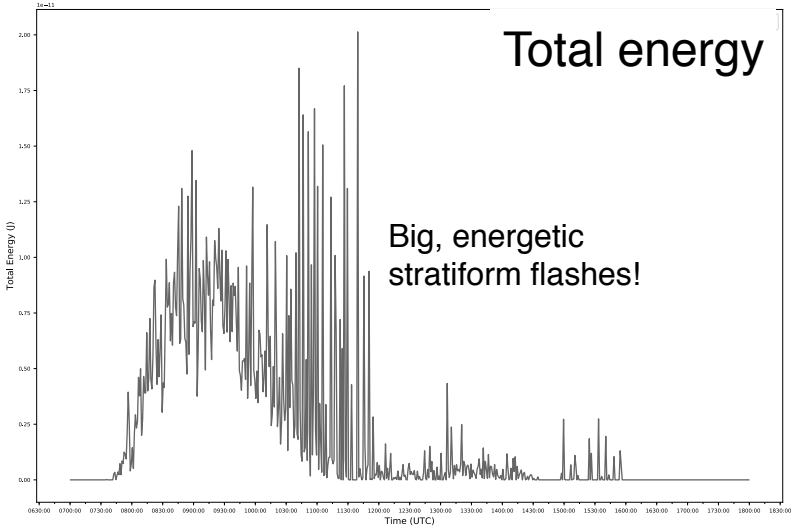

```
{
    "lassos": [
        {
            "created": "2017-09-06T14:44:45.0",
            "level": "INFO",
            "poly": {
                "filename": "None",
                "frame_id": 0,
                "frame_time": "2017-07-31T07:00:00",
                "frame_end": "2017-07-31T18:00:00",
                "field": "None",
                "title": "Lasso Verts",
                "x_verts": [
                    -102.38146385886124,
                    -102.38146385886124,
                    -101.30331460113878,
                    -101.30331460113878,
                    -102.38146385886124
                ],
                "y_verts": [
                    33.20895902214679,
                    34.11053353785319,
                    34.11053353785319,
                    33.20895902214679,
                    33.20895902214679
                ]
            }
        }
    ]
}
```


Bounding box spec file

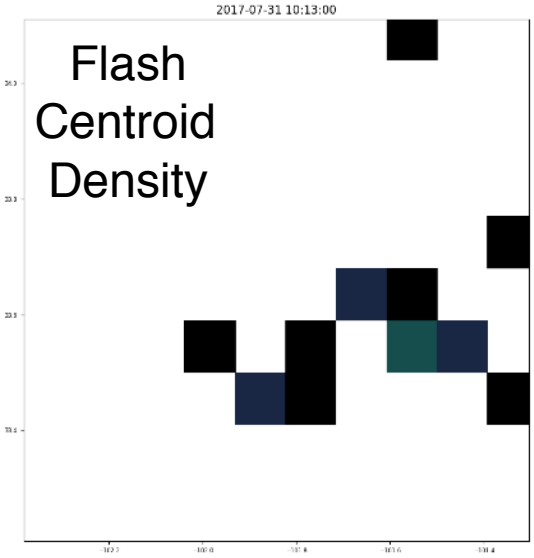
Flash centroid colored by time

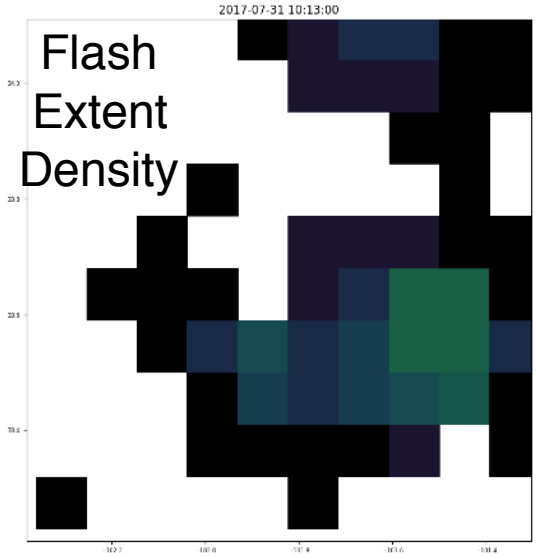


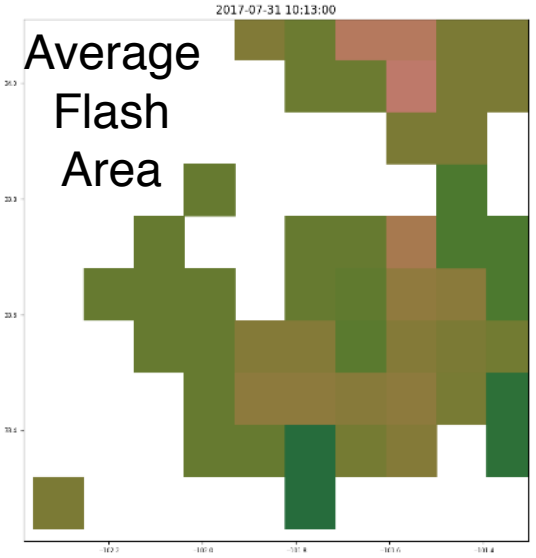




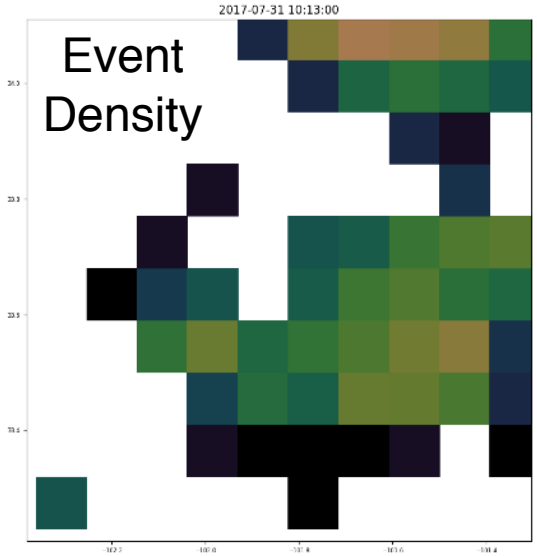
SAMPLE GRIDS

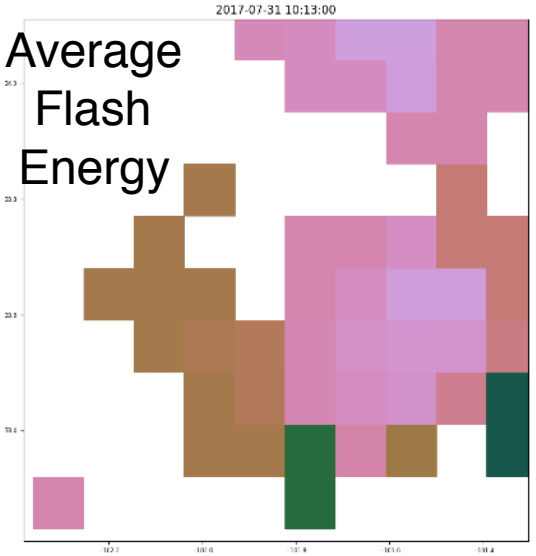






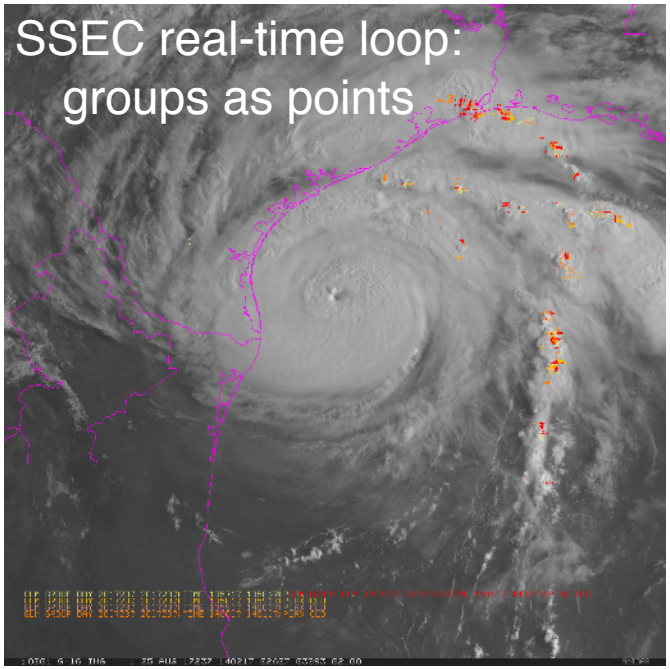
- Grids are subset and plotted within the bounding box used to calculate flash size statistics
- CSV of min/max/ percentiles of pixel statistics

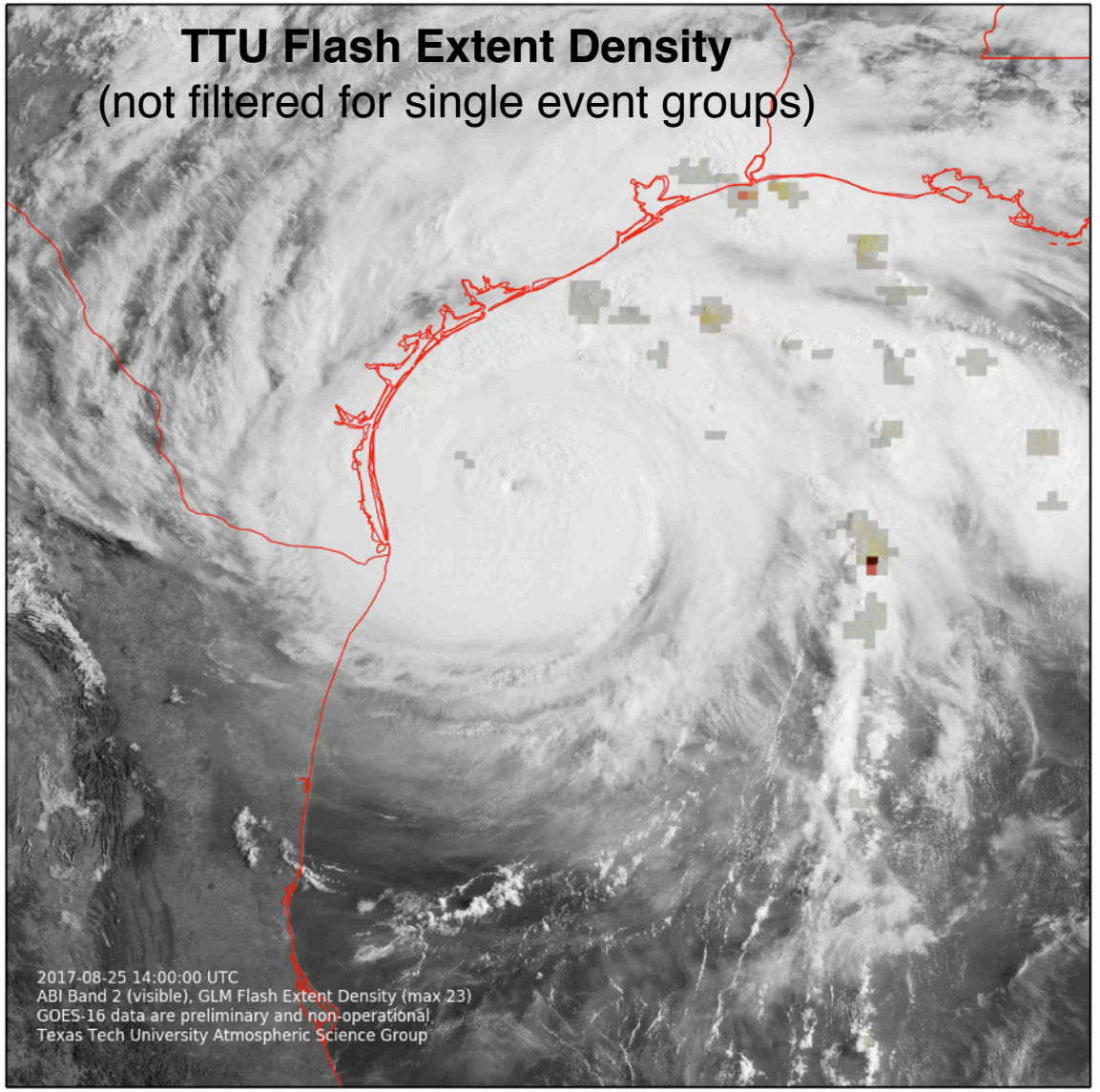




Harvey flash extent density & ABI meso sector (data via Unidata GRB / Amazon S3 test server)

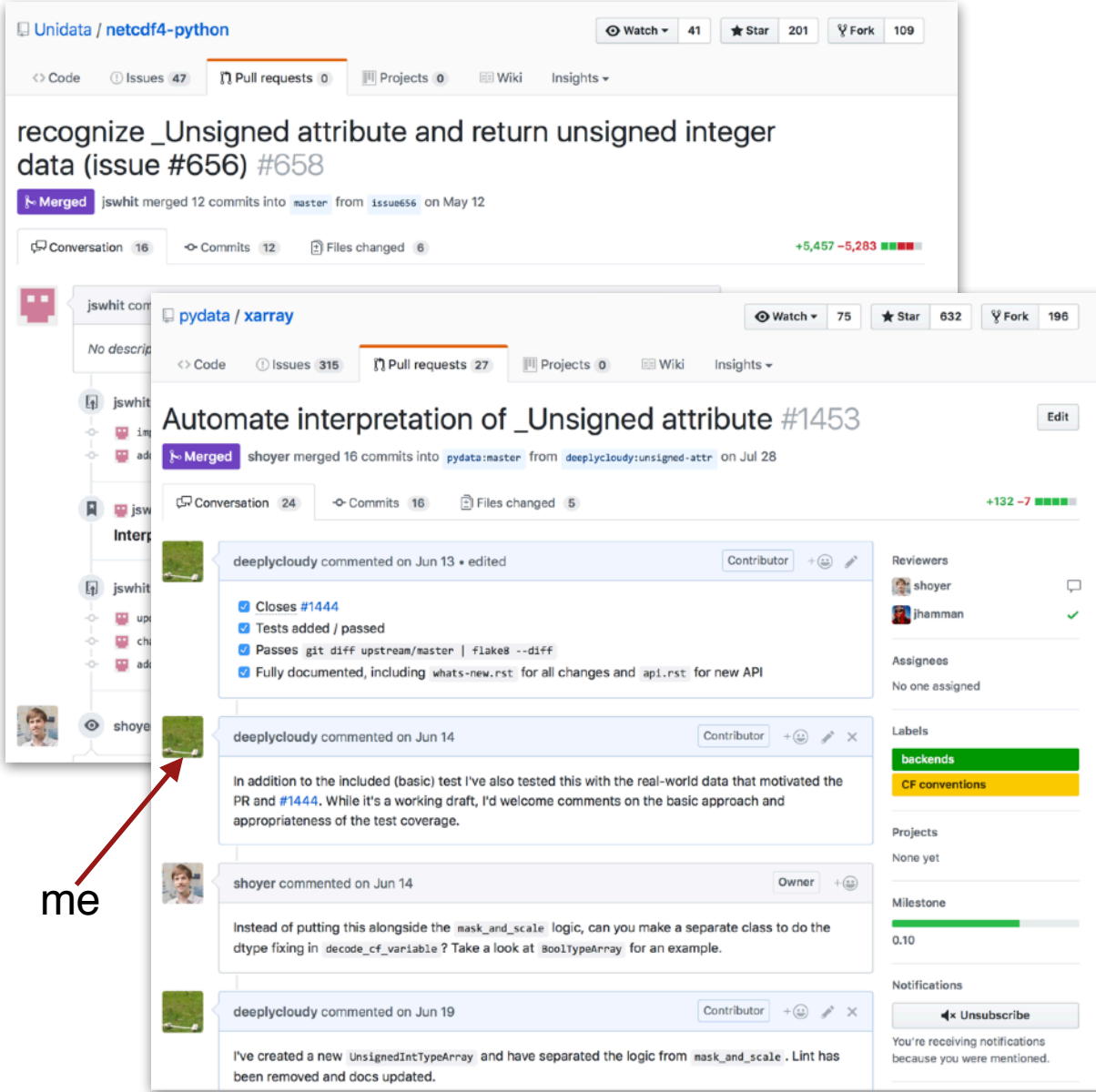
- Deep convection with mixed phase region easily distinguished from larger, less frequent extensive flashes
- Wide adoption of lightning data requires a reference for best practices to draw out these meteorological signals



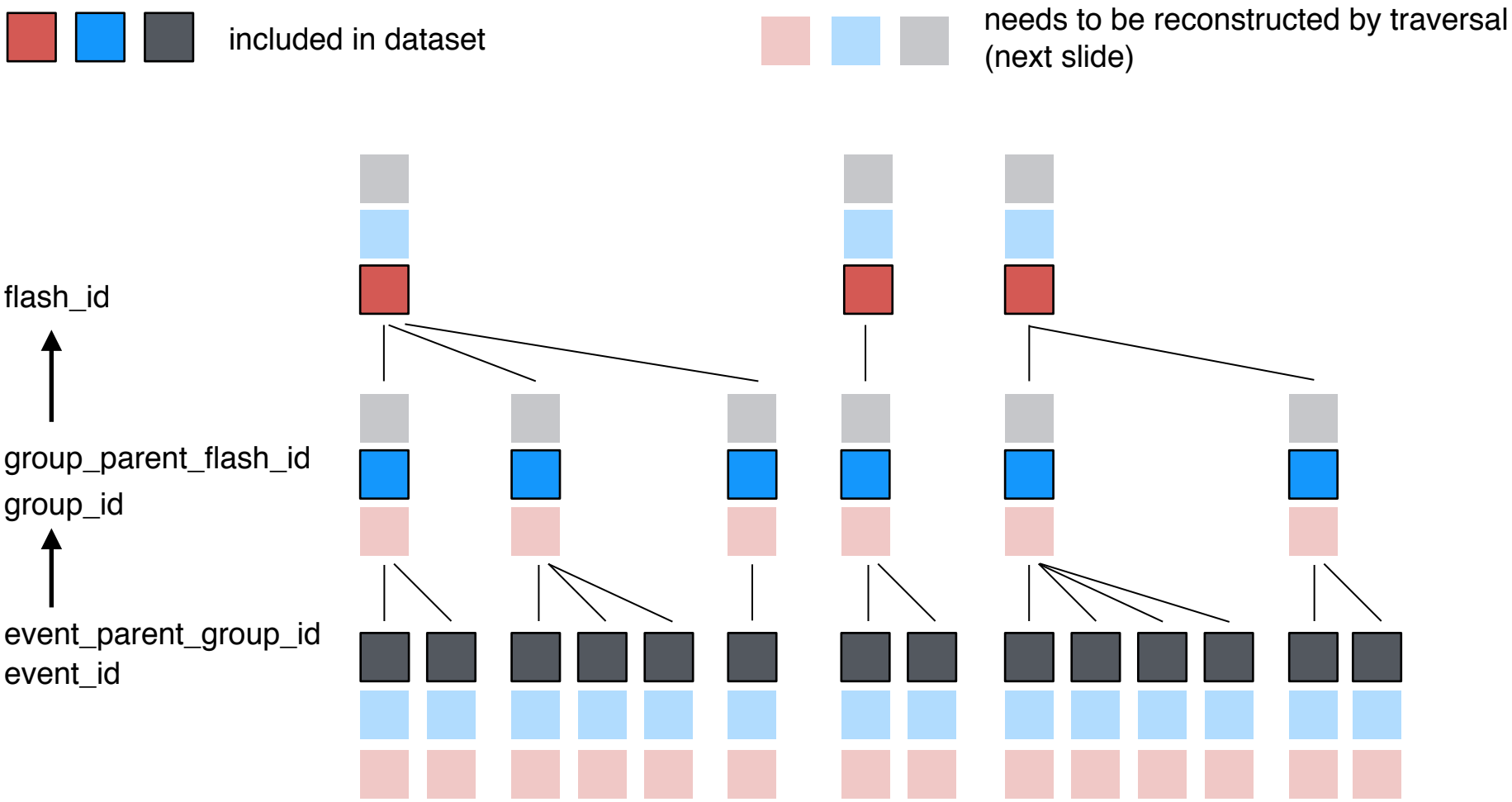


Along the way: Python NetCDF libraries now automate _Unsigned conversion

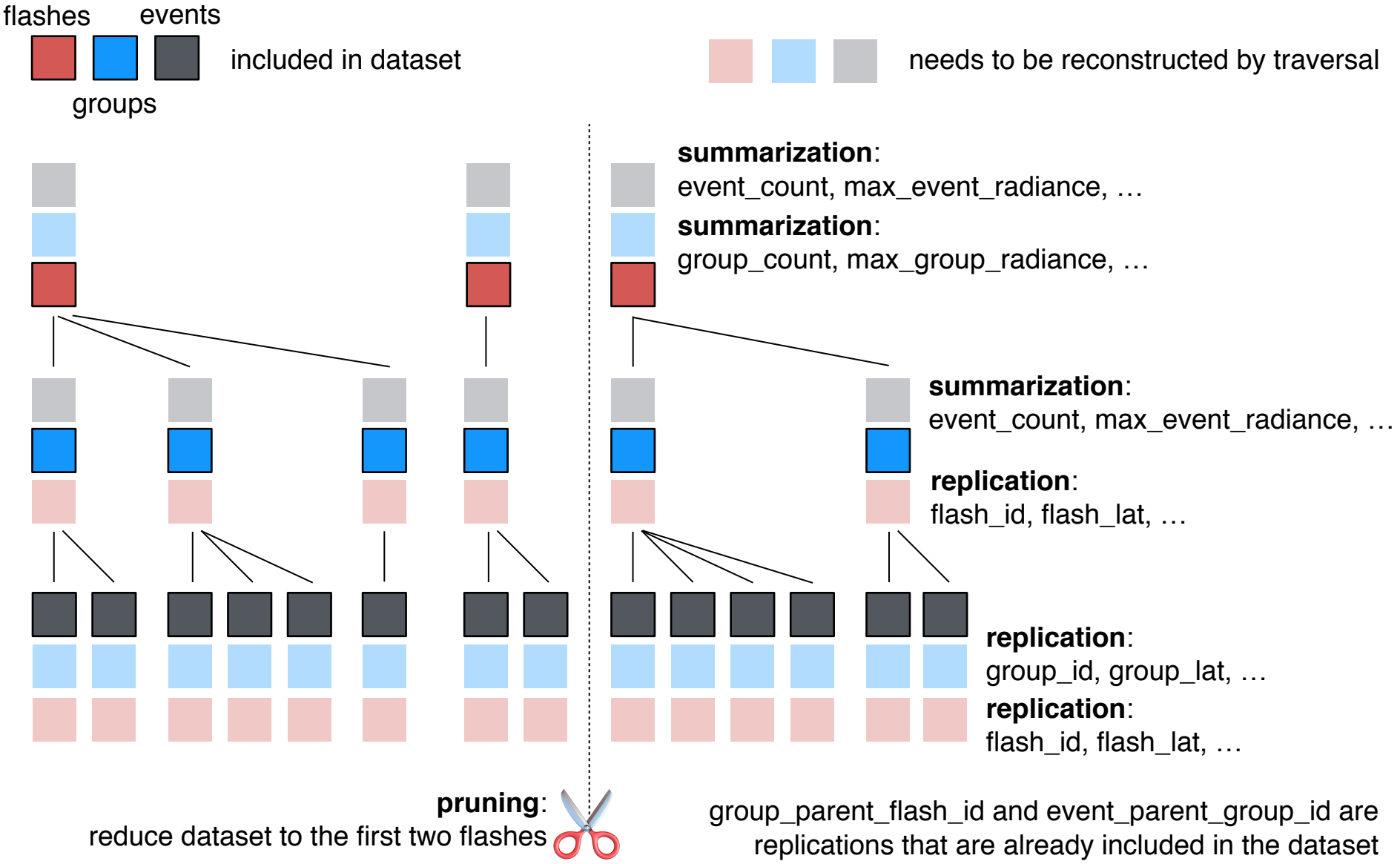
- NetCDF4-Python and xarray libraries
  - *Patches on 12 May and 28 July 2017*
- Both understand CF metadata conventions and automatically apply scale, and offset, and perform time conversion from NetCDF to Python-native types
- Patch threads contain some interesting history concerning the `_Unsigned` attribute from the maintainers of the NetCDF-4 libraries
- **Result: reading files "just works"**
  - As long as file metadata are correct



Full use of GLM data requires using the Hierarchical linkage among dimensions



GOAL: AUTOMATE THE TRAVERSAL OF ARBITRARILY MANY LEVELS OF THE HIERARCHY



Result: Generic traversal algorithm for hierarchically clustered parent-child data

- "Missing" GLM variables and operations are particular cases of the general traversal operation
  - `event_parent_flash_id` ← *Automatically calculated*
  - `flash_child_event_count` ← *when a GLM file in read*
  - `get_flashes_groups_events(flash_ids)`
- Extensible to arbitrary levels of hierarchy
  - *e.g., LIS Areas, storm cells identified by radar,*
  - *GLM-associated NLDN flash and its child strokes, etc.*
- Parent-child relationships are familiar as a foreign key schema in databases, but such ideas are not supported by meteorological dataset standards.
- `xarray`'s dimension-aware indexing and `groupby` functionality can mimic database-like behavior

Example: retrieve data for the 6 s flash near Lubbock, TX on 5 July 2017

```
from datetime import datetime
from glmtools.io.GLM import GLMDataset

glm = GLMDataset(filename)
flash_data = glm.subset_flashes(lon_range = (-102.5, -100.93),
                                lat_range = (32.5, 34.5))

group_time = flash_data.group_time_offset
big_flash_duration = np.asarray(
                         (datetime(2017,7,5,4,21,19),
                          datetime(2017,7,5,4,21,26))).astype('datetime64[ns]')
big_flash_sel = ( (group_time >= big_flash_duration[0]) &
                  (group_time <= big_flash_duration[1]) )

flash_ids = np.unique(flash_data.group_parent_flash_id[big_flash_sel])
flash_data = glm.get_flashes(flash_ids)
print(flash_data)
```

Open GLM L2 file and auto-calculate “missing” parent-child data

lat, lon subset (min events, groups, too)

Index group times with human-friendly `datetime`

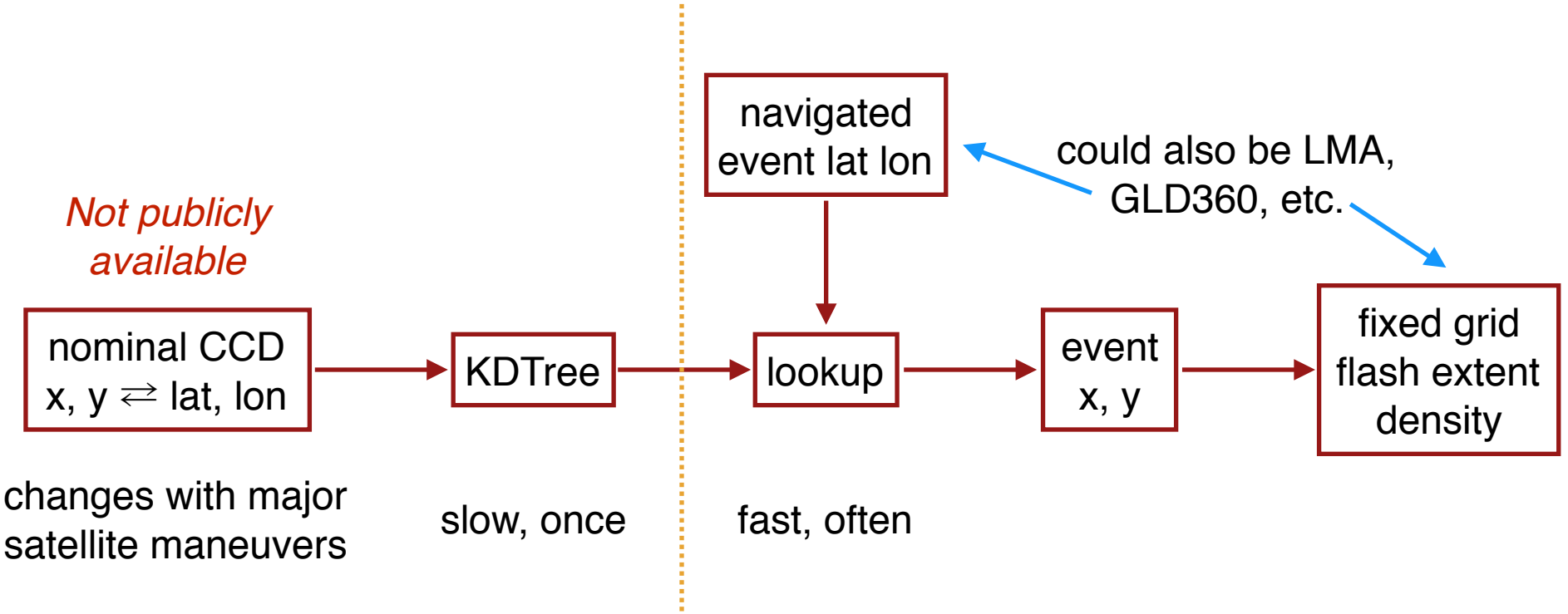
Find flash IDs

Get FGE data for these flashes

```
<xarray.Dataset>
Dimensions:                                (number_of_events: 8349, number_of_field_of_view_bounds: 2, number_of_flashes: 307, number_of_groups: 2043,
number_of_time_bounds: 2, number_of_wavelength_bounds: 2)
Coordinates:
  * number_of_flashes                      (number_of_flashes) MultiIndex
  - flash_id                               (number_of_flashes) int64 51893 ...
  - flash_time_offset_of_first_event       (number_of_flashes) datetime64[ns] 2017-07-05T04:21:19.912000 ...
  - flash_time_offset_of_last_event        (number_of_flashes) datetime64[ns] 2017-07-05T04:21:20.082000 ...
  - flash_lat                              (number_of_flashes) float64 33.48 ...
  - flash_lon                              (number_of_flashes) float64 -101.5 ...
  * number_of_groups                       (number_of_groups) MultiIndex
  - group_parent_flash_id                  (number_of_groups) int64 51893 ...
  - group_id                               (number_of_groups) int64 1000669504 ...
```

One approach for reconstruction of pixel ID information

- Pixel ID information not supplied in products
  - *Needed for operations: want to avoid choosing 8 vs. 9 vs. 10 km grid resolution*
- Can be reconstructed if nominal fixed grid is known
- KDTree allows for definition of arbitrarily stretched grid, including a GLM fixed grid matched to the CCD spacing



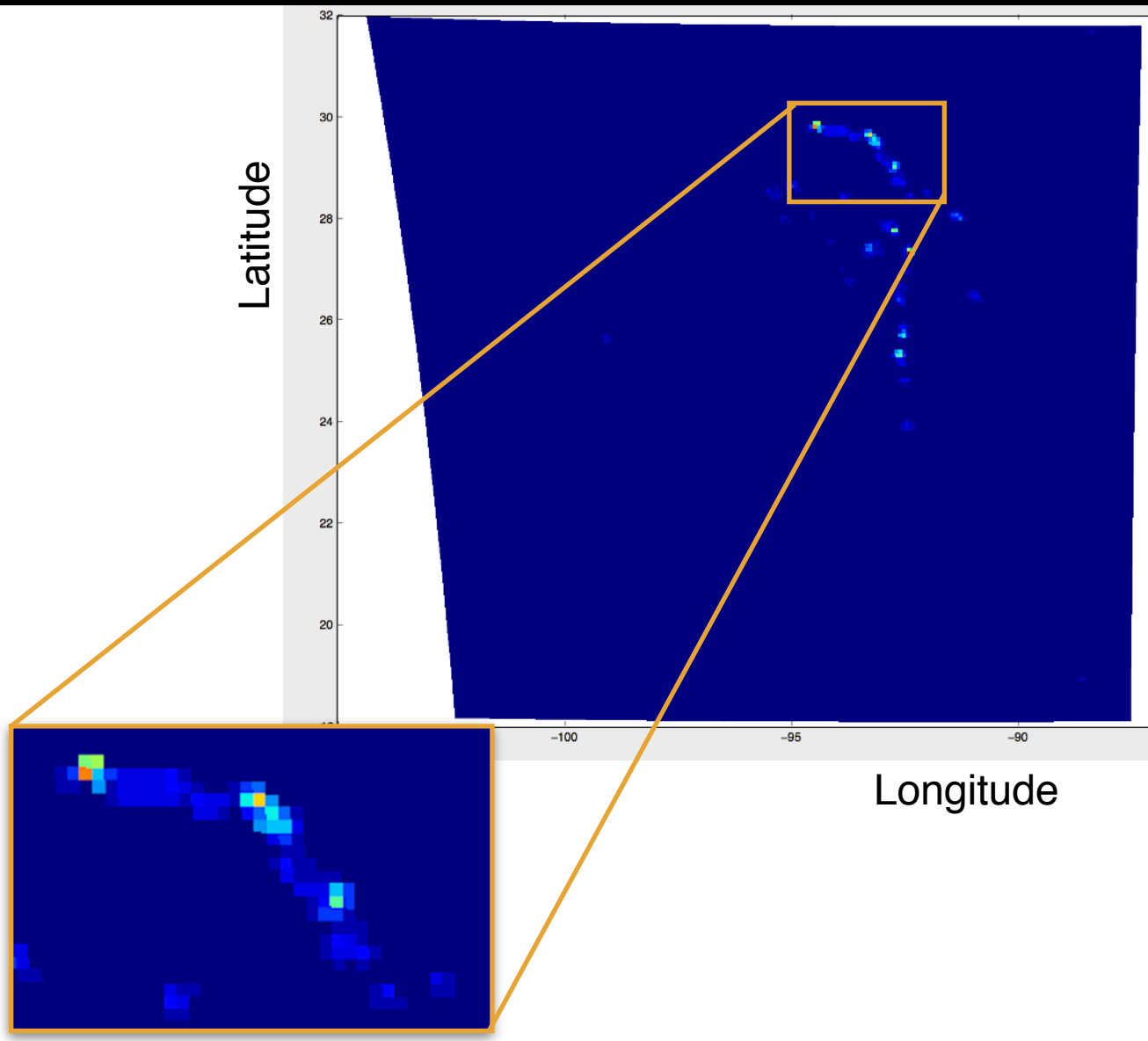
Fixed grid prototype for GLM data

- Supports arbitrary coordinate transforms among CCD xy ⇄ lat,lon,alt

  ⇄ ECEF cartesian

  ⇄ map projection

- Hierarchical traversal plus x,y lookup gives flash extent density on fixed grid
- Implemented as a command line switch in general-purpose gridding script



Future plans

- Working prototype of fixed grid suitable for NWS and science needs
- Package implements and automates many of the common operations needed for cal/val, science, R3, R20, etc, and can be used for bulk processing
- Intend to open-source these tools by the AMS annual meeting in January
- Private repository (unless you tell me to open-source it now), but I'm glad to give you access. Send me your GitHub username.
- A secondary goal of this tool is to support the long game: how do we encourage (correct, quality-controlled) usage of lightning data by non-specialists?